Microsoft Knowledge Base Article - 246335

HOWTO: Transfer Data from an ADO Recordset to Excel with Automation

Applies To

This article was previously published under Q246335

SUMMARY

You can transfer the contents of an ADO recordset to a Microsoft Excel worksheet by automating Excel. The approach that you can use depends on the version of Excel you are automating. Excel 97, Excel 2000, and Excel 2002 have a CopyFromRecordset method that you can use to transfer a recordset to a range. CopyFromRecordset in Excel 2000 and 2002 can be used to copy either a DAO or an ADO recordset. However, CopyFromRecordset in Excel 97 supports only DAO recordsets. To transfer an ADO recordset to Excel 97, you can create an array from the recordset and then populate a range with the contents of that array.

This article discusses both approaches. The sample code presented illustrates how you can transfer an ADO recordset to Excel 97, Excel 2000, or Excel 2002.

MORE INFORMATION

The code sample provided below shows how to copy an ADO recordset to a Microsoft Excel worksheet using automation from Microsoft Visual Basic. The code first checks the version of Excel. If Excel 2000 or 2002 is detected, the CopyFromRecordset method is used because it is efficient and requires less code. However, if Excel 97 or earlier is detected, the recordset is first copied to an array using the GetRows method of the ADO recordset object. The array is then transposed so that records are in the first dimension (in rows), and fields are in the second dimension (in columns). Then, the array is copied to an Excel worksheet through assigning the array to a range of cells. (The array is copied in one step rather than looping through each cell in the worksheet.)

The code sample uses the Northwind sample database that is included with Microsoft Office. If you selected the default folder when you installed Microsoft Office, the database is located in:

\Program Files\Microsoft Office\Office\Samples\Northwind.mdb

If the Northwind database is located in a different folder on your computer, you need to edit the path of the database in the code provided below.

If you do not have the Northwind database installed on your system, you can use the Add/Remove option for Microsoft Office setup to install the sample databases.

Steps to Create Sample

1. Start Visual Basic and create a new Standard EXE project. Form1 is created by default.
2. Add a **CommandButton** to Form1.
3. Click **References** from the **Project** menu. Add a reference to the **Microsoft ActiveX Data Objects 2.1 Library**.
4. Paste the following code into the code section of Form1:

```
Private Sub Command1_Click()
    Dim cnt As New ADODB.Connection
    Dim rst As New ADODB.Recordset

    Dim xlApp As Object
    Dim xlWb As Object
    Dim xlWs As Object


    Dim recArray As Variant

    Dim strDB As String
Dim fldCount As Integer
Dim recCount As Long
Dim iCol As Integer
Dim iRow As Integer

' Set the string to the path of your Northwind database
strDB = "c:\program files\Microsoft office\office11\samples\Northwind.mdb"

' Open connection to the database
cnt.Open "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=" & strDB & ";"

' Open recordset based on Orders table
rst.Open "Select * From Orders", cnt

' Create an instance of Excel and add a workbook
Set xlApp = CreateObject("Excel.Application")
Set xlWb = xlApp.Workbooks.Add
Set xlWs = xlWb.Worksheets("Sheet1")

' Display Excel and give user control of Excel's lifetime
xlApp.Visible = True
xlApp.UserControl = True

' Copy field names to the first row of the worksheet
fldCount = rst.Fields.Count
For iCol = 1 To fldCount
    xlWs.Cells(1, iCol).Value = rst.Fields(iCol - 1).Name
Next

' Check version of Excel
If Val(Mid(xlApp.Version, 1, InStr(1, xlApp.Version, ".") - 1)) > 8 Then
    'EXCEL 2000 or 2002: Use CopyFromRecordset

    ' Copy the recordset to the worksheet, starting in cell A2
    xlWs.Cells(2, 1).CopyFromRecordset rst
    'Note: CopyFromRecordset will fail if the recordset
    'contains an OLE object field or array data such
    'as hierarchical recordsets

Else
    'EXCEL 97 or earlier: Use GetRows then copy array to Excel

    ' Copy recordset to an array
    recArray = rst.GetRows
    'Note: GetRows returns a 0-based array where the first
    'dimension contains fields and the second dimension
    'contains records. We will transpose this array so that
    'the first dimension contains records, allowing the
    'data to appears properly when copied to Excel

    ' Determine number of records

    recCount = UBound(recArray, 2) + 1 '+ 1 since 0-based array


    ' Check the array for contents that are not valid when
    ' copying the array to an Excel worksheet
    For iCol = 0 To fldCount - 1
        For iRow = 0 To recCount - 1
            ' Take care of Date fields
            If IsDate(recArray(iCol, iRow)) Then
                recArray(iCol, iRow) = Format(recArray(iCol, iRow))
                ' Take care of OLE object fields or array fields
                ElseIf IsArray(recArray(iCol, iRow)) Then
                    recArray(iCol, iRow) = "Array Field"
                End If
            Next iRow 'next record
        Next iCol 'next field

        ' Transpose and Copy the array to the worksheet,
        ' starting in cell A2
        xlWs.Cells(2, 1).Resize(recCount, fldCount).Value = _
            TransposeDim(recArray)
    End If

    ' Auto-fit the column widths and row heights
    xlApp.Selection.CurrentRegion.Columns.AutoFit
    xlApp.Selection.CurrentRegion.Rows.AutoFit

    ' Close ADO objects
    rst.Close
    cnt.Close
    Set rst = Nothing
    Set cnt = Nothing

    ' Release Excel references
    Set xlWs = Nothing
    Set xlWb = Nothing

    Set xlApp = Nothing

End Sub


Function TransposeDim(v As Variant) As Variant
' Custom Function to Transpose a 0-based array (v)

    Dim X As Long, Y As Long, Xupper As Long, Yupper As Long
    Dim tempArray As Variant

    Xupper = UBound(v, 2)
    Yupper = UBound(v, 1)

    ReDim tempArray(Xupper, Yupper)
    For X = 0 To Xupper
        For Y = 0 To Yupper
            tempArray(X, Y) = v(Y, X)
        Next Y
    Next X

    TransposeDim = tempArray

End Function
```

5. Press the F5 key to run the project. Form1 appears.
6. Click the **CommandButton** on Form1, and note that the contents of the Orders table is displayed in a new workbook in Excel.

Using CopyFromRecordset

For efficiency and performance, CopyFromRecordset is the preferred method. Because Excel 97 supports only DAO recordsets with CopyFromRecordset, if you attempt to pass an ADO recordset to CopyFromRecordset with Excel 97, you receive the following error:

```
Run-time error 430:
Class does not support Automation or does not support expected interface.
```

In the code sample, you can avoid this error by checking Excel's version so that you do not use CopyFromRecordset for the 97 version.

Note When using CopyFromRecordset, you should be aware that the ADO or DAO recordset you use cannot contain OLE object fields or array data such as hierarchical recordsets. If you include fields of either type in a recordset, the CopyFromRecordset method fails with the following error:

```
Run-time error -2147467259:
Method CopyFromRecordset of object Range failed.
```

Using GetRows

If Excel 97 is detected, use the GetRows method of the ADO recordset to copy the recordset into an array. If you assign the array returned by GetRows to a range of cells in the worksheet, the data goes across the columns instead of down the rows. For example, if the recordset has two fields and 10 rows, the array appears as two rows and 10 columns. Therefore, you need to transpose the array using your TransposeDim() function before assigning the array to the range of cells. When assigning an array to a range of cells, there are some limitations to be aware of:

The following limitations apply when assigning an array to an Excel Range object:

- The array cannot contain OLE object fields or array data, such as hierarchical recordsets. Notice that the code sample checks for this condition and displays "Array Field" so that the user is made aware that the field cannot be displayed in Excel.

- The array cannot contain Date fields that have a date prior to the year 1900. (See the "References" section for a Microsoft Knowledge Base article link.) Note that the code sample formats Date fields as variant strings to avoid this potential problem.

Note the use of the TransposeDim() function to transpose the array before the array is copied to the Excel worksheet. Instead of creating your own function to transpose the array, you can use Excel's Transpose function by modifying the sample code to assign the array to the cells as shown below:

```
xlWs.Cells(2, 1).Resize(recCount, fldCount).Value = _
   xlApp.WorksheetFunction.Transpose(recArray)
```

If you decide to use Excel's Transpose method instead of the TransposeDim() function to transpose the array, you should be aware of the following limitations with the Transpose method:

- The array cannot contain an element that is greater than 255 characters.
- The array cannot contain Null values.
- The number of elements in the array cannot exceed 5461.

If the above limitations are not taken into consideration when you copy an array to an Excel worksheet, one of the following run-time errors may occur:

```
Run-time Error 13: Type Mismatch


Run-time Error 5: Invalid procedure call or argument


Run-time Error 1004: Application defined or object defined error
```

REFERENCES

For additional information about limitations on passing arrays to various versions of Excel, click the following article number to view the article in the Microsoft Knowledge Base:

177991 XL: Limitations of Passing Arrays to Excel Using Automation

For additional information, click the article numbers below to view the articles in the Microsoft Knowledge Base:

146406 XL: How to Retrieve a Table from Access into Excel Using DAO

215965 XL2000: 12:00:00 AM Displayed for Dates Earlier Than 1900

243394 HOWTO: Use MFC to Copy a DAO Recordset to Excel with Automation

247412 INFO: Methods for Transferring Data to Excel from Visual Basic

The information in this article applies to:

- Microsoft Office Excel 2003
- Microsoft Excel 2002
- Microsoft Excel 2000
- Microsoft Excel 97 for Windows
- Microsoft Visual Basic Professional Edition for Windows 5.0
- Microsoft Visual Basic Professional Edition for Windows 6.0
- Microsoft Visual Basic Enterprise Edition for Windows 5.0
- Microsoft Visual Basic Enterprise Edition for Windows 6.0
- ActiveX Data Objects (ADO) 2.0
- ActiveX Data Objects (ADO) 2.1
- ActiveX Data Objects (ADO) 2.5

Last Reviewed: 12/12/2003 (4.0)
Keywords: kbAutomation kbhowto KB246335

Send Print Help